Rocky Enterprise Linux 9.2 Manual Pages on command 'file.1'

C:\>man file.1

FILE(1) BSD General Commands Manual FILE(1)

NAME

file ? determine file type

SYNOPSIS

file [-bcdEhiklLNnprsSvzZ0] [--apple] [--extension] [--mime-encoding] [--mime-type]

[-e testname] [-F separator] [-f namefile] [-m magicfiles] [-P name=value]

file ...

file -C [-m magicfiles]

file [--help]

DESCRIPTION

This manual page documents version 5.38 of the file command.

file tests each argument in an attempt to classify it. There are three sets of

tests, performed in this order: filesystem tests, magic tests, and language tests.

The first test that succeeds causes the file type to be printed.

The type printed will usually contain one of the words text (the file contains only

printing characters and a few common control characters and is probably safe to read

on an ASCII terminal), executable (the file contains the result of compiling a pro?

gram in a form understandable to some UNIX kernel or another), or data meaning any?

thing else (data is usually ?binary? or non-printable). Exceptions are well-known

file formats (core files, tar archives) that are known to contain binary data. When

adding local definitions to /etc/magic, make sure to preserve these keywords. Users

depend on knowing that all the readable files in a directory have the word ?text?

printed. Don't do as Berkeley did and change ?shell commands text? to ?shell script?.

The filesystem tests are based on examining the return from a stat(2) system call. The program checks to see if the file is empty, or if it's some sort of special file. Any known file types appropriate to the system you are running on (sockets, symbolic links, or named pipes (FIFOs) on those systems that implement them) are intuited if they are defined in the system header file <sys/stat.h>.

The magic tests are used to check for files with data in particular fixed formats. The canonical example of this is a binary executable (compiled program) a.out file, whose format is defined in <elf.h>, <a.out.h> and possibly <exec.h> in the standard include directory. These files have a ?magic number? stored in a particular place near the beginning of the file that tells the UNIX operating system that the file is a binary executable, and which of several types thereof. The concept of a ?magic? has been applied by extension to data files. Any file with some invariant identifier at a small fixed offset into the file can usually be described in this way. The in? formation identifying these files is read from /etc/magic and the compiled magic file /usr/share/misc/magic.mgc, or the files in the directory /usr/share/misc/magic if the compiled file does not exist. In addition, if $HOME/.magic.mgc or $HOME/.magic ex? ists, it will be used in preference to the system magic files.

If a file does not match any of the entries in the magic file, it is examined to see if it seems to be a text file. ASCII, ISO-8859-x, non-ISO 8-bit extended-ASCII char? acter sets (such as those used on Macintosh and IBM PC systems), UTF-8-encoded Uni? code, UTF-16-encoded Unicode, and EBCDIC character sets can be distinguished by the different ranges and sequences of bytes that constitute printable text in each set. If a file passes any of these tests, its character set is reported. ASCII, ISO-8859-x, UTF-8, and extended-ASCII files are identified as ?text? because they will be mostly readable on nearly any terminal; UTF-16 and EBCDIC are only ?character data? because, while they contain text, it is text that will require translation be? fore it can be read. In addition, file will attempt to determine other characteris? tics of text-type files. If the lines of a file are terminated by CR, CRLF, or NEL, instead of the Unix-standard LF, this will be reported. Files that contain embedded escape sequences or overstriking will also be identified.

Once file has determined the character set used in a text-type file, it will attempt

to determine in what language the file is written. The language tests look for par? ticular strings (cf. <names.h>) that can appear anywhere in the first few blocks of a file. For example, the keyword .br indicates that the file is most likely a troff(1) input file, just as the keyword struct indicates a C program. These tests are less reliable than the previous two groups, so they are performed last. The lan? guage test routines also test for some miscellany (such as tar(1) archives, JSON files).

Any file that cannot be identified as having been written in any of the character sets listed above is simply said to be ?data?.

OPTIONS

--apple

Causes the file command to output the file type and creator code as used by older MacOS versions. The code consists of eight letters, the first describ? ing the file type, the latter the creator. This option works properly only for file formats that have the apple-style output defined.

-b, --brief

Do not prepend filenames to output lines (brief mode).

-C, --compile

Write a magic.mgc output file that contains a pre-parsed version of the magic file or directory.

-c, --checking-printout

Cause a checking printout of the parsed form of the magic file. This is usu? ally used in conjunction with the -m flag to debug a new magic file before installing it.

-d Prints internal debugging information to stderr.

-E On filesystem errors (file not found etc), instead of handling the error as regular output as POSIX mandates and keep going, issue an error message and exit.

-e, --exclude testname

Exclude the test named in testname from the list of tests made to determine the file type. Valid test names are:

apptype EMX application type (only on EMX).

ascii Various types of text files (this test will try to guess the text

encoding, irrespective of the setting of the ?encoding? option).

encoding Different text encodings for soft magic tests.

tokens Ignored for backwards compatibility.

cdf Prints details of Compound Document Files.

compress Checks for, and looks inside, compressed files.

csv Checks Comma Separated Value files.

elf Prints ELF file details, provided soft magic tests are enabled and the elf magic is found.

json Examines JSON (RFC-7159) files by parsing them for compliance.

soft Consults magic files.

tar Examines tar files by verifying the checksum of the 512 byte tar header. Excluding this test can provide more detailed content de? scription by using the soft magic method.

text A synonym for ?ascii?.

--extension

Print a slash-separated list of valid extensions for the file type found.

-F, --separator separator

Use the specified string as the separator between the filename and the file result returned. Defaults to ?:?.

-f, --files-from namefile

Read the names of the files to be examined from namefile (one per line) be? fore the argument list. Either namefile or at least one filename argument must be present; to test the standard input, use ?-? as a filename argument. Please note that namefile is unwrapped and the enclosed filenames are pro? cessed when this option is encountered and before any further options pro? cessing is done. This allows one to process multiple lists of files with different command line arguments on the same file invocation. Thus if you want to set the delimiter, you need to do it before you specify the list of files, like: ?-F @ -f namefile?, instead of: ?-f namefile -F @?.

-h, --no-dereference

option causes symlinks not to be followed (on systems that support symbolic links). This is the default if the environment variable POSIXLY_CORRECT is not defined.

-i, --mime

Causes the file command to output mime type strings rather than the more tra?
ditional human readable ones. Thus it may say ?text/plain; charset=us-ascii?
rather than ?ASCII text?.

--mime-type, --mime-encoding

Like -i, but print only the specified element(s).

-k, --keep-going

Don't stop at the first match, keep going. Subsequent matches will be have
the string ?\012- ? prepended. (If you want a newline, see the -r option.)
The magic pattern with the highest strength (see the -l option) comes first.

-l, --list

Shows a list of patterns and their strength sorted descending by magic(5)
strength which is used for the matching (see also the -k option).

-L, --dereference

option causes symlinks to be followed, as the like-named option in ls(1) (on
systems that support symbolic links). This is the default if the environment
variable POSIXLY_CORRECT is defined.

-m, --magic-file magicfiles

Specify an alternate list of files and directories containing magic. This
can be a single item, or a colon-separated list. If a compiled magic file is
found alongside a file or directory, it will be used instead.

-N, --no-pad

Don't pad filenames so that they align in the output.

-n, --no-buffer

Force stdout to be flushed after checking each file. This is only useful if
checking a list of files. It is intended to be used by programs that want
filetype output from a pipe.

-p, --preserve-date

On systems that support utime(3) or utimes(2), attempt to preserve the access
time of files analyzed, to pretend that file never read them.

-P, --parameter name=value

Set various parameter limits.

Name Default Explanation

indir 15 recursion limit for indirect magic

name 30 use count limit for name/use magic

elf_notes 256 max ELF notes processed

elf_phnum 128 max ELF program sections processed

elf_shnum 32768 max ELF sections processed

regex 8192 length limit for regex searches

bytes 1048576 max number of bytes to read from file

-r, --raw

Don't translate unprintable characters to \ooo. Normally file translates un? printable characters to their octal representation.

-s, --special-files

Normally, file only attempts to read and determine the type of argument files which stat(2) reports are ordinary files. This prevents problems, because reading special files may have peculiar consequences. Specifying the -s op? tion causes file to also read argument files which are block or character special files. This is useful for determining the filesystem types of the data in raw disk partitions, which are block special files. This option also causes file to disregard the file size as reported by stat(2) since on some systems it reports a zero size for raw disk partitions.

-S, --no-sandbox

On systems where libseccomp (https://github.com/seccomp/libseccomp) is avail? able, the -S flag disables sandboxing which is enabled by default. This op? tion is needed for file to execute external decompressing programs, i.e. when the -z flag is specified and the built-in decompressors are not available.

On systems where sandboxing is not available, this option has no effect.

Note: This Debian version of file was built without seccomp support, so this option has no effect.

-v, --version

Print the version of the program and exit.

-z, --uncompress

Try to look inside compressed files.

-Z, --uncompress-noreport

Try to look inside compressed files, but report information about the con?

tents only not the compression.

-0, --print0

Output a null character ?\0? after the end of the filename. Nice to cut(1) the output. This does not affect the separator, which is still printed.

If this option is repeated more than once, then file prints just the filename followed by a NUL followed by the description (or ERROR: text) followed by a second NUL for each entry.

--help Print a help message and exit.

ENVIRONMENT

The environment variable MAGIC can be used to set the default magic file name. If that variable is set, then file will not attempt to open $HOME/.magic. file adds ?.mgc? to the value of this variable as appropriate. The environment variable POSIXLY_CORRECT controls (on systems that support symbolic links), whether file will attempt to follow symlinks or not. If set, then file follows symlink, otherwise it does not. This is also controlled by the -L and -h options.

FILES

/usr/share/misc/magic.mgc Default compiled list of magic.

/usr/share/misc/magic Directory containing default magic files.

EXIT STATUS

file will exit with 0 if the operation was successful or >0 if an error was encoun? tered. The following errors cause diagnostic messages, but don't affect the program exit code (as POSIX requires), unless -E is specified:

? A file cannot be found

? There is no permission to read a file

? The file type cannot be determined

EXAMPLES

```
$ file file.c file /dev/{wd0a,hda}
file.c:   C program text
file:     ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
          dynamically linked (uses shared libs), stripped
/dev/wd0a: block special (0/0)
/dev/hda: block special (3/0)
$ file -s /dev/wd0{b,d}
```

/dev/wd0b: data

/dev/wd0d: x86 boot sector

$ file -s /dev/hda{,1,2,3,4,5,6,7,8,9,10}

/dev/hda: x86 boot sector

/dev/hda1: Linux/i386 ext2 filesystem

/dev/hda2: x86 boot sector

/dev/hda3: x86 boot sector, extended partition table

/dev/hda4: Linux/i386 ext2 filesystem

/dev/hda5: Linux/i386 swap file

/dev/hda6: Linux/i386 swap file

/dev/hda7: Linux/i386 swap file

/dev/hda8: Linux/i386 swap file

/dev/hda9: empty

/dev/hda10: empty

$ file -i file.c file /dev/{wd0a,hda}

file.c: text/x-c

file: application/x-executable

/dev/hda: application/x-not-regular-file

/dev/wd0a: application/x-not-regular-file

SEE ALSO

hexdump(1), od(1), strings(1), magic(5)

STANDARDS CONFORMANCE

This program is believed to exceed the System V Interface Definition of FILE(CMD), as near as one can determine from the vague language contained therein. Its behavior is mostly compatible with the System V program of the same name. This version knows more magic, however, so it will produce different (albeit more accurate) output in many cases.

The one significant difference between this version and System V is that this version treats any white space as a delimiter, so that spaces in pattern strings must be es? caped. For example,

>10 string language impress (imPRESS data)

in an existing magic file would have to be changed to

>10 string language\ impress (imPRESS data)

In addition, in this version, if a pattern string contains a backslash, it must be escaped. For example

```
0      string        \begindata     Andrew Toolkit document
```

in an existing magic file would have to be changed to

```
0      string        \\begindata    Andrew Toolkit document
```

SunOS releases 3.2 and later from Sun Microsystems include a file command derived from the System V one, but with some extensions. This version differs from Sun's only in minor ways. It includes the extension of the ?&? operator, used as, for ex? ample,

```
>16    long&0x7fffffff >0            not stripped
```

SECURITY

On systems where libseccomp (https://github.com/seccomp/libseccomp) is available, file is enforces limiting system calls to only the ones necessary for the operation of the program. This enforcement does not provide any security benefit when file is asked to decompress input files running external programs with the -z option. To en? able execution of external decompressors, one needs to disable sandboxing using the -S flag.

MAGIC DIRECTORY

The magic file entries have been collected from various sources, mainly USENET, and contributed by various authors. Christos Zoulas (address below) will collect addi? tional or corrected magic file entries. A consolidation of magic file entries will be distributed periodically.

The order of entries in the magic file is significant. Depending on what system you are using, the order that they are put together may be incorrect.

HISTORY

There has been a file command in every UNIX since at least Research Version 4 (man page dated November, 1973). The System V version introduced one significant major change: the external list of magic types. This slowed the program down slightly but made it a lot more flexible.

This program, based on the System V version, was written by Ian Darwin ?ian@darwinsys.com? without looking at anybody else's source code.

John Gilmore revised the code extensively, making it better than the first version.

Geoff Collyer found several inadequacies and provided some magic file entries. Con?

tributions of the ?&? operator by Rob McMahon, ?cudcv@warwick.ac.uk?, 1989.

Guy Harris, ?guy@netapp.com?, made many changes from 1993 to the present.

Primary development and maintenance from 1990 to the present by Christos Zoulas ?christos@astron.com?.

Altered by Chris Lowth ?chris@lowth.com?, 2000: handle the -i option to output mime type strings, using an alternative magic file and internal logic.

Altered by Eric Fischer ?enf@pobox.com?, July, 2000, to identify character codes and attempt to identify the languages of non-ASCII files.

Altered by Reuben Thomas ?rrt@sc3d.org?, 2007-2011, to improve MIME support, merge MIME and non-MIME magic, support directories as well as files of magic, apply many bug fixes, update and fix a lot of magic, improve the build system, improve the documentation, and rewrite the Python bindings in pure Python.

The list of contributors to the ?magic? directory (magic files) is too long to include here. You know who you are; thank you. Many contributors are listed in the source files.

LEGAL NOTICE

The files tar.h and is_tar.c were written by John Gilmore from his public-domain tar(1) program, and are not covered by the above license.

BUGS

Please report bugs and send patches to the bug tracker at https://bugs.astron.com/ or the mailing list at ?file@astron.com? (visit https://mailman.astron.com/mailman/listinfo/file first to subscribe).

TODO

Fix output so that tests for MIME and APPLE flags are not needed all over the place, and actual output is only done in one place. This needs a design. Suggestion: push possible outputs on to a list, then pick the last-pushed (most specific, one hopes) value at the end, or use a default if the list is empty. This should not slow down evaluation.

The handling of MAGIC_CONTINUE and printing \012- between entries is clumsy and complicated; refactor and centralize.

Some of the encoding logic is hard-coded in encoding.c and can be moved to the magic files if we had a !:charset annotation

Continue to squash all magic bugs. See Debian BTS for a good source.

Store arbitrarily long strings, for example for %s patterns, so that they can be printed out. Fixes Debian bug #271672. This can be done by allocating strings in a string pool, storing the string pool at the end of the magic file and converting all the string pointers to relative offsets from the string pool.

Add syntax for relative offsets after current level (Debian bug #466037).

Make file -ki work, i.e. give multiple MIME types.

Add a zip library so we can peek inside Office2007 documents to print more details about their contents.

Add an option to print URLs for the sources of the file descriptions.

Combine script searches and add a way to map executable names to MIME types (e.g. have a magic value for !:mime which causes the resulting string to be looked up in a table). This would avoid adding the same magic repeatedly for each new hash-bang in? terpreter.

When a file descriptor is available, we can skip and adjust the buffer instead of the hacky buffer management we do now.

Fix ?name? and ?use? to check for consistency at compile time (duplicate ?name?, ?use? pointing to undefined ?name?). Make ?name? / ?use? more efficient by keeping a sorted list of names. Special-case ^ to flip endianness in the parser so that it does not have to be escaped, and document it.

If the offsets specified internally in the file exceed the buffer size (HOWMANY variable in file.h), then we don't seek to that offset, but we give up. It would be better if buffer managements was done when the file descriptor is available so move around the file. One must be careful though because this has performance (and thus security considerations).

AVAILABILITY

You can obtain the original author's latest version by anonymous FTP on ftp.astron.com in the directory /pub/file/file-X.YZ.tar.gz.